Philipps Universität Marburg

Übungen zu „Konzepte von Programmiersprachen“, WS 2010/11

Prof. Dr. R. Loogen · Fachbereich Mathematik und Informatik · Hans-Meerwein-Straße, D-35032 Marburg

Nr. 11, Abgabe: Dienstag, 18. Januar 2011 vor der Vorlesung

28. Prozessnetze — 3 Punkte

Definieren Sie als Prozessnetze:

(a) Die Liste aller Zweierpotenzen: $(2^n \mid n \geq 0)$. / 1

(b) Die geschachtelte Liste aller Binomialkoeffizienten $((\binom{n}{k} \mid 0 \leq k \leq n) \mid n \geq 0)$. / 2

29. Polynomfunktionen — 7 Punkte

Polynome in einer Variablen können in Haskell als unendliche Listen von Zahlen dargestellt werden: `type Poly = [Float]`.
Beispielsweise repräsentiert die Liste `(1:0:2:(-4):0:3:repeat 0)` das Polynom $1 + 2x^2 - 4x^3 + 3x^5$.
Definieren Sie die folgenden Operationen über Polynomen:

(a) `scale :: Float -> Poly -> Poly` zur Multiplikation mit einem Skalar: (b) `addPoly :: Poly -> Poly -> Poly` zur Addition von zwei Polynomen / 1+1

$$a * \sum_{i=0}^{\infty} b_i x^i = \sum_{i=0}^{\infty} a b_i x^i \qquad \sum_{i=0}^{\infty} a_i x^i + \sum_{i=0}^{\infty} b_i x^i = \sum_{i=0}^{\infty} (a_i + b_i) x^i$$

(c) `mulPoly :: Poly -> Poly -> Poly` zur Multiplikation von Polynomen / 2

$$\sum_{i=0}^{\infty} a_i x^i * \sum_{i=0}^{\infty} b_i x^i = a_0 * \sum_{i=0}^{\infty} b_i x^i + x * \sum_{i=0}^{\infty} a_{i+1} x^i * \sum_{i=0}^{\infty} b_i x^i$$

(d) `divPoly :: Poly -> Poly -> Poly` zur Division von Polynomen / 2

$$\frac{\sum_{i=0}^{\infty} a_i x^i}{\sum_{i=0}^{\infty} b_i x^i} = \frac{a_0}{b_0} + x * \frac{\sum_{i=0}^{\infty} \left(a_{i+1} - \frac{a_0}{b_0} b_{i+1}\right) x^i}{\sum_{i=0}^{\infty} b_i x^i}$$

(e) Berechnen Sie die Liste der Fibonacci-Zahlen durch Ausnutzung der folgenden Gleichung: $\frac{1}{1 - x - x^2} = 1 + x + 2x^2 + 3x^3 + 5x^4 + 8x^5 + \ldots$ / 1

30. Ein-/Ausgabe mit `interact` — 2 Punkte

(a) Definieren Sie die in der Vorlesung eingeführte Funktion / 1
`mapUntil :: (a -> Bool) -> (a -> a) -> [a] -> [a]`
(siehe Datei bspLaziness.hs auf der Vorlesungsseite) mit vordefinierten Funktionen höherer Ordnung.

(b) Schreiben Sie mit `mapUntil` und `interact :: (String -> String) -> IO ()` / 1
ein Programm, das wiederholt eine Eingabezeile in Großbuchstaben wieder ausgibt, bis die Eingabe einer Leerzeile erfolgt. Ergänzen Sie Ihr Programm um Eingabeaufforderungen und eine Schlussausgabe.